DOCKET NO.: 03-LJ-017
CLIENT NO.: STMI01-03017
Customer No.: 30425

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of	:	Suresh Rajgopal et al.
Serial No.	:	10/750,012
Filed	:	December 31, 2003
For	:	APPARATUS AND METHOD USING HASHING FOR EFFICIENTLY IMPLEMENTING AN IP LOOKUP SOLUTION IN HARDWARE
Art Unit	:	2419
Examiner	:	Bo Hui Alvin Zhu
Confirmation No.	:	9337

MAIL STOP AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Appellant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a notice of appeal. The review is requested for the reasons stated in the arguments below, demonstrating the clear legal and factual deficiency of the rejections of some or all claims.

Claims 1-3, 5-16 and 18-22 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,665,297 to *Hariguchi, et al* in view of U.S. Patent Application Publication No. 2001/0027479 to *Delaney, et al*. Claims 4 and 17 were rejected under 35 U.S.C. § 103(a) over *Hariguchi et al* in view of *Delaney et al* and further in view of U.S. Patent No. 6,625,612 to *Tal et al*.

Independent claims 1 and 10 claim each recite that each hast table is allocated a group of the memory blocks based on a size of the respective hash table. Similarly, independent claim 14 recites that a number of memory blocks allocated to a hash table is based on a size of the respective hash table. Such a feature is not found in the cited references. The cited portion of *Delaney et al* actually teaches the exact opposite of allocating memory blocks based on a size of the hash table, instead teaching that the hash table is sized to fit within a fixed size group of memory blocks:

> [0043] Each peer client maintains two hash-tables which contain information about data package location: a local-data packages table and a network-data packages table. The local-data packages table is a hash-table of data packages which reside on the storage medium or media of the peer client itself. The network-data packages table is a hash-table of data packages which reside on the storage medium or media of other clients on the local network. This table contains the local area network address of the peer client on which each data package is being stored. The size of this hash-table is preferably limited in order to reduce memory consumption. More preferably, each entry in the table has a time-stamp, such that <u>*older entries are purged when the size of the table exceeds the upper permissible limit*</u>.

Delaney et al, ¶ [0043] (emphasis added).

Claim 2 recites that each hash table is allocated a smallest number of memory blocks sufficient to hold prefixes for which no collision occurs within the respective hash table. Similarly, claim 15 recites maintaining each hash table within a smallest number of memory blocks sufficient

to hold all required prefixes for which no collision occurs within the respective hash table. Such features are not found in the cited references. The cited portions of *Hariguchi et al*, taken in context, merely describe *what* is stored in the routing table when and how routing table entries are removed, rather than allocating a smallest number of memory blocks to the hash table that is sufficient to hold the entries:

> Because the hash tables 70 (FIG. 2) match masked portions of the destination address, the hash tables will often have collisions when the tables are updated. *The CAM based routing table 80 is used to avoid hash collisions in the hash-based routing tables 70 by storing the destination address, prefix length and, in an alternate embodiment, the output pointer, when a hash table already has an entry with the same hash value as the destination address*: During searching, the hash tables do not have collisions and multiple matches or hits in the hash tables and CAM are resolved in a subsequent stage.
>
> . . . .
>
> When an entry is to be deleted, the destination address of the route entry is searched in the routing tables. The hash match register 171 of the priority encoder 170 is checked. The output pointer is removed from a hash bucket if the information in the hash match register 171 matches the information in the route entry to be removed. At the same time, the CAM match register 146 is checked. When the CAM hit/miss flag indicates a hit and unless the information in the hash match register 171 matches the information in the route entry to be removed, the route entry is removed from the CAM when the information in the CAM match register 146 matches the destination address for the associated prefix length.

Hariguchi et al, column 6, lines , column 9, lines 12-20 (emphasis added). The above-quote portions of *Hariguchi et al* say nothing of the size of routing tables or allocation of memory blocks to a routing table. *Hariguchi et al* contains no discussion of the allocating memory based on the size of the hash table to be stored therein, but instead teaches that a fixed size memory is employed with entry space reused for other CAM entries once the content of a particular entry is deleted:

> When an entry is deleted from a hash table, a CAM entry may be moved to that deleted entry in the hash table to enhance CAM utilization. If the CAM stores a

> destination address that can be stored in that hash table then that destination address is deleted from the CAM and stored in the hash table. In other words, an entry is moved from the CAM to the hash table if the hash value of the destination address of the entry in the CAM is equal to the index of the hash table in which the deleted address was stored.

Hariguchi et al, column 9, lines 35-43.

Claims 3, 12 and 16 each recite that the variable number of memory blocks allocated to a hash table is limited to a predetermined maximum number. Such a feature is not found in the cited references. The cited portion of *Delaney et al* teaches that the size of the hash table is limited to reduce memory consumption in storing that table (within a fixed amount of memory), not that the variable number of memory blocks assigned to a hash table is limited to a predetermined maximum. Claim 5 recites that each hash table contains different length prefixes. Claim 18 similarly recites storing, in each of a plurality of hash tables, prefixes of a different length than prefixes contained in any other of the plurality of hash tables. Such a feature is not found in the cited references. *Hariguchi et al*, cited in the Office Action as teaching this feature, actually only teaches that multiple hash tables 70 are used, and that separate hash circuits 82 are each dedicated to parallel searching of the hash tables based on a unique prefix length. *Hariguchi et al* does not teach that each hash table 70 stores only prefixes of a different length than those in the remaining hash tables, as suggested in the Office Action. Instead, use of multiple hash tables 70 in *Hariguchi et al* appears to be motivated by speeding up "pipelined" searching, such that each hash circuit searches the hash tables consecutively while the remaining hash circuits similarly search the other hash tables consecutively, with each hash table containing entries for any possible prefix length.

For these reasons, the Applicants assert that the claims in the application are in condition for allowance and that the rejection of the claims is both factually and legally deficient. The Applicants respectfully request this case be returned to the Examiner for allowance or, alternatively, further examination.

The Commissioner is hereby authorized to charge any additional fees connected with this communication or credit any overpayment to Deposit Account No. 50-0208.

Respectfully submitted,

MUNCK CARTER, LLP

Date: 11-17-09

Daniel E. Venglarik
Registration No. 39,409

P.O. Box 802432
Dallas, Texas 75380
(972) 628-3600 (main number)
(972) 628-3616 (fax)
E-mail: *dvenglarik@munckcarter.com*